



I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date indicated below.

By: Margus Wolff Date: November 12, 2003

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applic. No. : 10/662,627
Applicant : Astrid Elbe, et al.
Filed : September 15, 2003

Docket No. : S&ZIO020104
Customer No. : 24131

CLAIM FOR PRIORITY

Commissioner for Patents,
P.O. Box 1450, Alexandria, VA 22313-1450

Sir:

Claim is hereby made for a right of priority under Title 35, U.S. Code, Section 119, based upon the German Patent Application 101 11 987.9, filed March 13, 2001.

A certified copy of the above-mentioned foreign patent application is being submitted herewith.

Respectfully submitted,

Margus Wolff
For Applicant

Date: November 12, 2003

Lerner and Greenberg, P.A.
Post Office Box 2480
Hollywood, FL 33022-2480
Tel: (954) 925-1100
Fax: (954) 925-1101

/av



Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen: 101 11 987.9

Anmeldetag: 13. März 2001

Anmelder/Inhaber: Infineon Technologies AG,
München/DE

Bezeichnung: Verfahren und Vorrichtung zum modularen
Multiplizieren

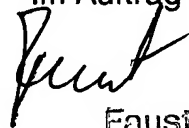
IPC: G 06 F 7/52

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 16. Oktober 2003
Deutsches Patent- und Markenamt

Der Präsident

Im Auftrag



Faust

Patentanwälte · Postfach 710867 · 81458 München
Infineon Technologies AG
St.-Martin-Str. 53

81669 München

PATENTANWÄLTE

European Patent Attorneys
European Trademark Attorneys

Fritz Schoppe, Dipl.-Ing.
Tankred Zimmermann, Dipl.-Ing.
Ferdinand Stöckeler, Dipl.-Ing.
Franz Zinkler, Dipl.-Ing.

Telefon/Telephone 089/790445-0
Telefax/Facsimile 089/790 22 15
Telefax/Facsimile 089/74996977
e-mail: szsz_iplaw@t-online.de

Verfahren und Vorrichtung zum modularen Multiplizieren

MIT AKTE

Beschreibung

Verfahren und Vorrichtung zum modularen Multiplizieren

5 Die vorliegende Erfindung bezieht sich auf Kryptographiealgorithmen und Vorrichtungen zum Ausführen solcher Kryptographiealgorithmen und insbesondere auf ein Verfahren und eine Vorrichtung zum modularen Multiplizieren unter Verwendung eines Multiplikations-Vorausschau-Verfahrens und eines Reduktions-Vorausschau-Verfahrens.
10

Die Kryptographie ist eine der wesentlichen Anwendungen für die modulare Arithmetik. Ein wesentlicher Algorithmus für die Kryptographie ist der bekannte RSA-Algorithmus. Der RSA-
15 Algorithmus baut auf einer modularen Exponentiation auf, welche folgendermaßen dargestellt werden kann:

$$C = M^d \bmod (N).$$

20 Hierbei ist C eine verschlüsselte Nachricht, M ist eine nicht-verschlüsselte Nachricht, d ist der geheime Schlüssel und N ist der Modul. Der Modul N wird üblicherweise durch Multiplikation zweier Primzahlen p und q erzeugt. Die modulare Exponentiation wird mittels des bekannten Square-and-Multiply-Algorithmus in Multiplikationen zerlegt. Hierzu wird
25 der Exponent d in Zweierpotenzen zerlegt, so daß die modulare Exponentiation in mehrere modulare Multiplikationen zerlegt werden kann. Um die modulare Exponentiation rechenmäßig effizient implementieren zu können, wird die modulare Exponentiation daher in modulare Multiplikationen zerlegt, welche dann
30 in modulare Additionen zerlegt werden können.

Die DE 3631992 C2 offenbart ein Kryptographie-Verfahren, bei dem die modulare Multiplikation unter Verwendung eines Multiplikations-Vorausschau-Verfahrens und unter Verwendung eines Reduktions-Vorausschau-Verfahrens beschleunigt werden kann.
35 Das in der DE 3631992 C2 beschriebene Verfahren wird auch als

ZDN-Verfahren bezeichnet und anhand von Fig. 9 näher beschrieben. Nach einem Startschritt 900 des Algorithmus werden die globalen Variablen M, C und N initialisiert. Ziel ist es, folgende modulare Multiplikation zu berechnen:

5

$$Z = M * C \bmod N.$$

M wird als der Multiplikator bezeichnet, während C als der Multiplikand bezeichnet wird. Z ist das Ergebnis der modularen Multiplikation, während N der Modul ist.

10

Hierauf werden verschiedene lokale Variablen initialisiert, auf die zunächst nicht näher eingegangen werden braucht. Anschließend werden zwei Vorausschau-Verfahren angewandt. Im Multiplikations-Vorausschau-Verfahren GEN_MULT_LA wird unter Verwendung verschiedener Look-Ahead-Regeln ein Multiplikations-Verschiebungswert s_z sowie ein Multiplikations-Vorausschau-Parameter a berechnet (910). Hierauf wird der gegenwärtige Inhalt des Z-Registers einer Links-Verschiebungs-Operation um s_z -Stellen unterzogen (920).

15

20

25

30

35

Im wesentlichen parallel dazu wird ein Reduktions-Vorausschau-Verfahren GEN_Mod_LA (930) durchgeführt, um einen Reduktionsverschiebungswert s_N und einen Reduktions-Parameter b zu berechnen. In einem Schritt 940 wird dann der gegenwärtige Inhalt des Modul-Registers, also N, um s_N Stellen nach links bzw. rechts verschoben, um einen verschobenen Modulwert N' zu erzeugen. Die zentrale Drei-Operanden-Operation des ZDN-Verfahrens findet in einem Schritt 950 statt. Hierbei wird das Zwischenergebnis Z' nach dem Schritt 920 zu dem Multiplikanden C, der mit dem Multiplikations-Vorausschau-Parameter a multipliziert ist, und zu dem verschobenen Modul N' , der mit dem Reduktions-Vorausschau-Parameter b multipliziert ist, addiert. Je nach aktueller Situation können die Vorausschau-Parameter a und b einen Wert von +1, 0 oder -1 haben.

Ein typischer Fall besteht darin, daß der Multiplikations-Vorausschau-Parameter a $+1$ beträgt, und daß der Reduktion-Vorausschau-Parameter b -1 beträgt, so daß zu einem verschobenen Zwischenergebnis Z' der Multiplikand C hinzu addiert wird, und der verschobene Modul N' davon subtrahiert wird. a wird einen Wert gleich 0 haben, wenn das Multiplikations-Vorausschau-Verfahren mehr als eine voreingestellte Anzahl von einzelnen Links-Verschiebungen zulassen würde, also wenn s_z größer als der maximal zulässige Wert von s_z ist, der auch als k bezeichnet wird. Für den Fall, daß a gleich 0 ist, und daß Z' aufgrund der vorausgehenden modularen Reduktion, also der vorausgehenden Subtraktion des verschobenen Moduls noch ziemlich klein ist, und insbesondere kleiner als der verschobene Modul N' ist, muß keine Reduktion stattfinden, so daß der Parameter b gleich 0 ist.

Die Schritte 910 bis 950 werden so lange durchgeführt, bis sämtliche Stellen des Multiplikanden abgearbeitet sind, also bis m gleich 0 ist, und bis auch ein Parameter n gleich 0 ist, welcher angibt, ob der verschobene Modul N' noch größer als der ursprüngliche Modul N ist, oder ob trotz der Tatsache, daß bereits sämtliche Stellen des Multiplikanden abgearbeitet sind, noch weitere Reduktionsschritte durch Subtrahieren des Moduls von Z durchgeführt werden müssen.

Abschließend wird noch bestimmt, ob Z kleiner als 0 ist. Falls dies der Fall ist, muß, um eine abschließende Reduktion zu erreichen, der Modul N zu Z hinzuaddiert werden, damit schließlich das korrekte Ergebnis Z der modularen Multiplikation erhalten wird. In einem Schritt 960 ist die modulare Multiplikation mittels des ZDN-Verfahrens beendet.

Der Multiplikations-Verschiebungswert s_z sowie der Multiplikations-Parameter a , welche im Schritt 910 durch den Multiplikations-Vorausschau-Algorithmus berechnet werden, ergeben sich durch die Topologie des Multiplikators sowie durch die

eingesetzten Vorausschau-Regeln, die in der DE 3631992 C2 beschrieben sind.

Der Reduktions-Verschiebungswert s_N und der Reduktions-

5 Parameter b werden, wie es ebenfalls in der DE 3631992 C2 beschrieben ist, durch Vergleich des gegenwärtigen Inhalts des Z-Registers mit einem Wert $2/3$ mal N bestimmt. Aufgrund dieses Vergleiches trägt das ZDN-Verfahren seinen Namen (ZDN = Zwei Drittel N).

10

Das ZDN-Verfahren, wie es in Fig. 9 dargestellt ist, führt die modulare Multiplikation auf eine Drei-Operanden-Addition (Block 950 in Fig. 9) zurück, wobei zur Steigerung der Rechenzeiteffizienz das Multiplikations-Vorausschau-Verfahren und damit einhergehend das Reduktions-Vorausschau-Verfahren eingesetzt werden. Im Vergleich zur Montgomery-Reduktion für kann daher ein Rechenzeitvorteil erreicht werden.

15

Im nachfolgenden wird anhand von Fig. 10 näher auf das Reduktions-Vorausschau-Verfahren eingegangen, das im Block 930 von Fig. 9 ausgeführt wird. Zunächst wird in einem Block 1000 eine Reservierung für die lokalen Variablen, d. h. den Reduktions-Vorausschau-Parameter b und den Reduktions-Verschiebungswert s_N , durchgeführt. In einem Block 1010 wird der Reduktions-Verschiebungswert s_N auf Null initialisiert. Hierauf wird in einem Block 1020 der Wert ZDN berechnet, der gleich $2/3$ des Moduls N ist. Dieser Wert, der im Block 1020 bestimmt wird, wird in einem eigenen Register, dem ZDN-Register, auf dem Kryptocoprozessor abgespeichert.

25

30

In einem Block 1030 wird dann bestimmt, ob die Variable n gleich 0 ist, oder ob der Verschiebungswert s_N gleich $-k$ ist. k ist ein Wert, welcher den maximalen Verschiebungswert, welcher durch die Hardware vorgegeben ist, definiert. Im ersten Durchgang wird der Block 1030 mit NEIN beantwortet, so daß in einem Block 1040 der Parameter n dekrementiert wird, und daß in einem Block 1060 auch der Reduktions-Verschiebungswert um

35

1 dekrementiert wird. Dann wird in einem Block 1080 die Variable ZDN neu belegt, nämlich mit ihrem halben Wert, was durch eine Rechts-Verschiebung des im ZDN-Register stehenden Werts ohne weiteres erreicht werden kann. In einem Block 1100
5 wird dann festgestellt, ob der Absolutwert des aktuellen Zwischenergebnisses größer als der im ZDN-Register stehende Wert ist.

Diese Vergleichsoperation im Block 1100 ist die zentrale Operation des Reduktions-Vorausschau-Verfahrens. Wird die Frage
10 mit JA beantwortet, so ist die Iteration beendet und der Reduktions-Vorausschau-Parameter b wird, wie es im Block 1120 dargestellt ist, belegt. Wird die im Block 1100 zu beantwortende Frage dagegen mit NEIN beantwortet, so wird iterativ
15 wieder zurückgesprungen, um die aktuellen Werte von n und s_N im Block 1030 zu untersuchen. Wird der Block 1030 irgendwann in der Iteration mit JA beantwortet, so wird zu einem Block 1140 gesprungen, in dem der Reduktions-Parameter b zu Null gesetzt wird. In der in Fig. 9 im Block 950 dargestellten
20 Drei-Operanden-Operation führt dies dazu, daß kein Modul addiert oder subtrahiert wird, was bedeutet, daß das Zwischenergebnis Z so klein war, daß keine modulare Reduktion erforderlich war. In einem Block 1160 wird dann die Variable n neu belegt, wobei dann in einem Block 1180 schließlich der Reduk-
25 tions-Verschiebungs-Wert s_N berechnet wird, welcher in einem Block 940 von Fig. 9 benötigt wird, um die Linksverschiebung des Moduls durchzuführen, um einen verschobenen Modul zu erreichen.

30 In den Blöcken 1200, 1220 und 1240 werden schließlich die aktuellen Werte von n und k hinsichtlich weiterer Variablen MAX und cur_k untersucht, um die aktuelle Belegung des N-Registers zu untersuchen, um sicherzustellen, daß keine Registerüberschreitungen stattfinden. Die näheren Details sind
35 für die vorliegende Erfindung nicht von Bedeutung, sind jedoch in der DE 3631992 C2 detailliert beschrieben.

Der in den Fig. 9 und 10 dargestellte Algorithmus kann hardwaremäßig implementiert werden, wie es in Fig. 7 dargestellt ist. Für die in dem Block 950 durchzuführende Drei-Operanden-Operation wird eine arithmetische Einheit 700 benötigt, die in Fig. 7 mit AU bezeichnet ist. Dieselbe ist mit einem Register C 710 für den Multiplikanden, mit einem Register N 720 für den Modul und mit einem Register Z 730 für das aktuelle Zwischenergebnis der modularen Multiplikation gekoppelt. Aus Fig. 7 ist ferner zu sehen, daß das Ergebnis der Drei-Operanden-Operation über einen Rückkopplungspfeil 740 wieder in das Z-Register 730 eingespeist wird. Aus Fig. 7 ist ferner die Verbindung der Register untereinander zu sehen. Der in dem Block 1020 von Fig. 10 berechnete Wert ZDN muß in einem eigenen ZDN-Register 750 abgespeichert werden. Der ZDN-Vergleich bzw. die in Fig. 10 dargestellte Iterationschleife wird ferner durch eine eigene Steuerlogik 760 für den ZDN-Vergleich ablaufmäßig gesteuert.

Die Hauptarbeit des ZDN-Algorithmus zur Berechnung von $Z := M \times C \bmod N$ besteht daher in den folgenden beiden Operationen:

1. Berechnung der Verschiebungswerte s_z und s_i für die Register Z und N, so daß folgende Gleichung erfüllt ist:

$$2/3 N \times 2^{-s_i} < |Z| \leq 4/3 N \times 2^{-s_i} \quad \text{und}$$

2. Berechnung der Drei-Operanden-Summe:

$$Z := 2^{s_z} Z + a C + b \times 2^{s_z - s_i} N,$$

Der Multiplikations-Vorausschau-Parameter a und der Reduktions-Vorausschau-Parameter b können, wie es bekannt ist, Werte von -1, 0 und +1 einnehmen.

Es sei darauf hingewiesen, daß das Zwischenergebnis Z, der Multiplikand C und der Modul N Langzahlen sind, also Zahlen, deren Anzahl von Stellen bzw. Bits durchaus größer als 512

sein können, wobei diese Zahlen auch bis zu über 2048 Stellen haben können.

Der in dem Block 1100 durchzuführende Vergleich des aktuellen
5 Zwischenergebnisses Z mit dem Wert ZDN wird jedoch aus Re-
chenzeitgründen nicht mit sämtlichen Bits von Z durchgeführt,
sondern lediglich mit einer Anzahl von höchstwertigen Bits
von Z, wobei sich hierfür eine Anzahl von 32 Bits als ausrei-
chend herausgestellt hat, um eine sehr hohe Genauigkeit für
10 das Vergleichsergebnis zu erhalten.

Für die für diesen Vergleich erforderlichen 32 höchstwertigen
Bits von $2/3 N$ wird, wie es in Fig. 7 durch das Bezugszeichen
750 dargestellt ist, ein eigenes Register benötigt, welches
15 als das ZDN-Register bezeichnet wird.

Ferner wird ein eigener Hardware-Komparator benötigt, der für
den aktuellen Wert im Z-Register und den aktuellen Wert im
ZDN-Register den richtigen s_i -Wert berechnet, so daß folgende
20 Gleichung erfüllt ist:

$$2/3 \cdot 2^{-s_i} N < |Z| \leq 4/3 \cdot 2^{-s_i} N$$

Nachteilig an diesem Verfahren ist daher zum einen, daß so-
25 wohl das zusätzliche ZDN-Register als auch der Hardware-
Komparator extra Chipfläche benötigen. Andererseits ist die
Berechnung von $2/3 N$ und die Berechnung des Hilfs-
Verschiebungswerts s_i im ZDN-Algorithmus, die durch die in
Fig. 10 gezeigte Iterationsschleife durchgeführt wird, für
30 den gesamten Algorithmus zeitkritisch und kann durchaus be-
stimmend für die Gesamtausführungszeit des Algorithmus sein.

Die Aufgabe der vorliegenden Erfindung besteht darin, ein
verbessertes Konzept zum modularen Multiplizieren zu schaf-
35 fen, das zum einen platzsparender implementiert werden kann,
und das zum anderen weniger Rechenzeit benötigt.

Diese Aufgabe wird durch ein Verfahren zum modularen Multiplizieren nach Patentanspruch 1 oder durch eine Vorrichtung zum modularen Multiplizieren nach Patentanspruch 14 gelöst.

5 Der vorliegenden Erfindung liegt die Erkenntnis zugrunde, daß der rechenzeitaufwendige Vergleich des aktualisierten Zwischenergebnisses mit dem Wert ZDN , also dem $2/3$ -fachen des Moduls N , dadurch erleichtert werden kann, wenn zunächst der Modul N in einen transformierten Modul N^T transformiert wird,
10 und die gesamte modulare Multiplikation mit dem transformierten Modul N^T anstatt des eigentlichen Moduls durchgeführt wird. Erfindungsgemäß wird der Modul derart transformiert, daß der vorbestimmte Bruchteil des transformierten Moduls, also bei einem bevorzugten Ausführungsbeispiel das $2/3$ -fache des transformierten Moduls eine bestimmte Zahl wird, die so
15 gewählt wird, daß der Vergleich von $2/3 N^T$ mit dem Zwischenergebnis Z trivial wird. Gemäß der vorliegenden Erfindung wird die Transformation so durchgeführt, daß der vorbestimmte Bruchteil des transformierten Moduls eine höherwertige Stelle mit einem ersten vorbestimmten Wert hat, der zumindest eine
20 niederwertige Stelle folgt, die einen zweiten vorbestimmten Wert hat. In binärer Darstellung und Zweier-Komplement-Konvention, in der das höchstwertige Bit das Vorzeichen angibt, wird die Transformation des Moduls in den transformierten Modul so ausgeführt, daß das zweit-höchstwertige Bit von
25 $2/3 N^T$ eine binäre Eins ist, während das dritthöchste Bit und noch weitere niederwertigere Bits Nullen sind.

In diesem Fall ist der Vergleich derart trivial, daß einfach
30 die Anzahl der Stellen zwischen der höchstwertigen Eins des vorbestimmten Bruchteils des transformierten Moduls und dem aktualisierten Zwischenergebnis Z der modularen Multiplikation abgezählt werden muß, um den Verschiebungswert s_i zu erhalten, aus dem dann der Reduktions-Verschiebungswert s_N einfach
35 dadurch ermittelt werden kann, daß von dem Multiplikations-Verschiebungs-Wert des parallel ablaufenden Multiplikations-Vorausschau-Verfahrens der sogenannte Hilfs-

Verschiebungswert s_i , welcher durch den ZDN-Vergleich erhalten wird, subtrahiert wird.

Das gesamte ZDN-Verfahren wird genauso wie im Stand der Technik abgearbeitet. Es wird jedoch nun statt des Moduls N der transformierte Modul N^T verwendet, so daß schließlich ein „Transformations-Ergebnis“ der modularen Multiplikation erhalten wird, das in der Restklasse des transformierten Moduls N^T ist. Eine abschließende Rücktransformation, derart, daß das Transformations-Ergebnis der modularen Multiplikation unter Verwendung des ursprünglichen Moduls N modular reduziert wird, liefert dann das eigentliche Ergebnis der modularen Multiplikation des Multiplikators M mit dem Multiplikanden C unter Verwendung des Moduls N .

Bevorzugte Ausführungsbeispiele der vorliegenden Erfindung werden nachfolgend beziehungsweise auf die beiliegenden Zeichnungen detailliert erläutert. Es zeigen:

Fig. 1 ein Ablaufdiagramm des erfindungsgemäßen Konzepts zum modularen Multiplizieren;

Fig. 2 die Aufteilung eines Moduls N in einen ersten Abschnitt N_T von Bits und in einen zweiten Abschnitt N_R von Bits;

Fig. 3 die Aufteilung des transformierten Moduls N^T in einen ersten Abschnitt von Stellen mit der Länge L (N^T) und die verbleibenden Stellen;

Fig. 4 eine Darstellung der Stellen des 2/3-fachen des transformierten Moduls N^T ;

Fig. 5 eine schematische Darstellung der Stellen des transformierten Moduls mit Randomisierung;

Fig. 6 eine schematische Darstellung eines Rechenwerks zum Ausführen der modularen Multiplikation gemäß der vorliegenden Erfindung;

5 Fig. 7 eine schematische Darstellung eines Rechenwerks für das bekannte ZDN-Verfahren;

Fig. 8a bis 8c eine schematische Darstellung des Zusammenhangs zwischen dem Multiplikations-
 10 Verschiebungswert s_z , dem Hilfs-Verschiebungswert s_i und dem Reduktions-Verschiebungswert s_N ;

Fig. 9 eine Flußdiagrammdarstellung des bekannten ZDN-Verfahrens; und

15

Fig. 10 eine Flußdiagrammdarstellung des bekannten Reduktions-Vorausschau-Verfahrens.

Fig. 1 zeigt ein Ablaufdiagramm des erfindungsgemäßen Verfahrens zum modularen Multiplizieren eines Multiplikanden C mit
 20 einem Multiplikator M unter Verwendung eines Moduls N. Zunächst wird in einem Schritt 10 der Modul N in einen transformierten Modul N^T gemäß folgender Gleichung transformiert:

25
$$N^T = T \times N.$$

In einem Schritt 12 wird dann die modulare Multiplikation unter Verwendung des transformierten Moduls N^T und des vorbestimmten Bruchteils des transformierten Moduls, der beim bevorzugten Ausführungsbeispiel 2/3 beträgt, abgearbeitet. Bezogen auf die modulare Exponentiation bedeutet dies, daß eine
 30 RSA-Gleichung folgender Form berechnet wird:

$$C^T = M^d \bmod N^T.$$

35

Es wird also das Ergebnis der modularen Exponentiation C nicht in der durch den Modul N definierten Restklasse sondern

in der durch den transformierten Modul N^T definierten Restklasse berechnet, weshalb auf der linken Seite der obigen Gleichung nicht C sondern C^T steht. Das erfindungsgemäße Konzept zeichnet sich dadurch aus, daß durch die Verwendung des transformierten Moduls N^T die Berechnung des Hilfs-
5 Reduktions-Verschiebungswerts s_i , die der Iterationsschleife von Fig. 10 des bekannten Reduktions-Vorausschau-Verfahrens entspricht, stark vereinfacht ist.

- 10 In einem abschließenden Schritt 14 wird dann wieder eine Rück-Transformation von N^T zu N durchgeführt, indem eine Operation ausgeführt wird, die folgender Gleichung entspricht:

$$C: = C^T \bmod N.$$

15

Das transformierte Ergebnis C^T , das in der Restklasse des transformierten Moduls N^T liegt, wird dabei vorzugsweise durch eine einfache Verschiebungs/Subtraktions-Reduktion in die Restklasse des Moduls N zurückgeführt, so daß C das Ergebnis der modularen Exponentiation ist.
20

Die Transformation des Moduls N in einen transformierten Modul N^T unter Verwendung des Transformators T aus Schritt 10 wird so durchgeführt, daß der vorbestimmte Bruchteil des transformierten Moduls, also beim bevorzugten Ausführungsbeispiel das 2/3-fache des transformierten Moduls, eine höherwertige Stelle mit einem ersten vorbestimmten Wert hat, der zumindest eine niederwertige Stelle folgt, die einen zweiten vorbestimmten Wert hat. Damit kann der Vergleich des Zwischen-
25 schenergebnisses Z mit dem 2/3-fachen des transformierten Moduls stark vereinfacht werden, nämlich indem die oberste Stelle von Z , die ebenfalls den ersten vorbestimmten Wert hat, gesucht wird, und die Differenz zwischen der höherwertigen Stelle mit ersten vorbestimmten Wert des vorbestimmten
30 Bruchteils des transformierten Moduls und der obersten Stelle des Zwischen-
35 schenergebnisses Z mit dem ersten vorbestimmten Wert gleich der Differenz s_i ist.

Zusammengefaßt stellt sich dies folgendermaßen dar. N wird vorzugsweise in der 32-Bit-CPU und nicht im Krypto-Coprozessor in einen transformierten Modul N^T transformiert, so daß gilt:

$$N^T = T \times N,$$

wobei T eine natürliche Zahl ist.

Für N^T ergibt sich folgende Gestalt, wenn sämtliche verwendeten Zahlen Binärzahlen sind:

$$N^T = 1100... 0 \text{ XX...XX}$$

Für das 2/3-fache des transformierten Moduls ergibt sich dann folgender Wert:

$$2/3 N^T = 100... 0 \text{ X'X'...X'X'}$$

Aus N^T und $2/3 N^T$ ist zu sehen, daß beide eine erste Portion von beispielsweise 16 Bits haben, und dann eine Portion von $L(N)$ Bits X bzw. X'. Für den sogenannten ZDN-Vergleich werden nur die obersten 16 Bits des 2/3-fachen des transformierten Moduls N^T herangezogen, da sich dann bereits eine Fehlerwahrscheinlichkeit von besser als etwa 2^{-10} ergibt. Es müssen also nicht alle 512, 1024 oder 2048 Bits des 2/3-fachen des transformierten Moduls zum ZDN-Vergleich herangezogen werden, sondern es genügt, wenn dieser Vergleich mit den obersten 16 Bits des transformierten Moduls durchgeführt wird. Selbstverständlich könnten auch noch weniger Bits von $2/3 N^T$ zum Vergleich herangezogen werden, dann steigt jedoch die Fehlerwahrscheinlichkeit nach und nach an. Da die Fehler jedoch unkritisch sind und nur zu einem suboptimalen Verhalten des Reduktions-Vorausschau-Verfahrens führen, ist dieser Weg ohne weiteres gangbar.

Das $2/3$ -fache des transformierten Moduls N^T hat somit eine höherwertige Stelle mit dem Wert 1, der zumindest eine niederwertige Stelle folgt, die einen Wert 0 hat, also einen zweiten vorbestimmten Wert. Bei dem vorstehend beschriebenen Ausführungsbeispiel ist die Anzahl der niederwertigen Stellen 15. Selbstverständlich können auch hier größere oder kleinere Anzahlen genommen werden, je nach dem, welche Größenunterschiede zwischen dem Zwischenergebnis Z und dem $2/3$ -fachen des transformierten Moduls N^T zu erwarten sind bzw. bearbeitet werden sollen. Für den Betrag des Zwischenergebnisses Z der modularen Multiplikation, also des Ergebnisses der Drei-Operanden-Addition im Block 950 von Fig. 9 ergibt sich folgende Gestalt:

$$|Z| = 00\dots 01YY\dots Y$$

Der Hilfs-Verschiebungswert s_i wird gemäß folgender Gleichung berechnet:

$$2/3 N^T \times 2^{-s_i} < |Z| \leq 4/3 N^T \times 2^{-s_i}.$$

Aufgrund der Topologie des $2/3$ -fachen des transformierten Moduls N^T ist der Wert s_i immer der Abstand zwischen dem höchstwertigen Bit mit einer 1 des $2/3$ -fachen des transformierten Moduls N^T und der höchstwertigen 1 des Betrags des Zwischenergebnisses.

Erfindungsgemäß kann diese Stellendifferenz bzw. der Wert s_i trivial ermittelt werden. Keine Iteration ist mehr erforderlich.

Darüber hinaus ist kein ZDN-Register mehr erforderlich, um das $2/3$ -fache des Moduls zu speichern, da per Definition zumindest die oberen beispielsweise 16 Bit des $2/3$ -fachen des transformierten Moduls N^T immer die gleiche Gestalt haben. Kein Bit-Komparator ist mehr erforderlich. Die Wertigkeitsdifferenz der höchstwertigen Stelle des $2/3$ -fachen des trans-

formierten Moduls N^T mit einer „1“ und der höchstwertigen Stelle von Z mit einer „1“ kann ohne weiteres beispielsweise durch eine bitweise XOR-Verknüpfung des Registers für den transformierten Modul und des Registers für das Zwischenergebnis Z durchgeführt werden. s_i ist dann gleich der Differenz der Wertigkeit der Stelle, wo die XOR-Verknüpfung eine erste „1“ ausgibt, und wo die XOR-Verknüpfung eine zweite „1“ ausgibt.

10 Aufgrund der Tatsache, daß kein ZDN-Register und kein ZDN-Komparator erforderlich sind, ist das gesamte Rechenwerk auf einer kleineren Chipfläche unterzubringen.

Außerdem hat der Krypto-Control-Part, also die Steuerlogik für den ZDN-Vergleich (760 in Fig. 7), eine kleinere Komplexität, da die aufwendige Iterationsschleife von Fig. 10 nicht ausgeführt werden muß. Schließlich geht die Berechnung schneller, so daß sich durch die Berechnung des Hilfs-Verschiebungswerts s_i keine Timing-Probleme mehr für den gesamten Algorithmus ergeben.

Im nachfolgenden wird anhand der Figuren 2 bis 5 auf die erfindungsgemäße Transformation genauer eingegangen.

25 Wie es bereits ausgeführt worden ist, besteht ein wesentlicher Teil des ZDN-Algorithmus darin, daß folgende Gleichung erfüllt ist

$$2/3 \cdot 2^{-s_i} N < |Z| \leq 4/3 \cdot 2^{-s_i} N.$$

30

s_i wird als Hilfs-Verschiebungswert bezeichnet und ist der Verschiebungswert, der notwendig ist, um Z stellenmäßig zu derselben Position wie N zu schieben. Im Stand der Technik waren zur Berechnung von s_i Vergleichsoperationen von $|Z|$ mit $2/3 N$ notwendig.

35

Erfindungsgemäß wird der Vergleich mit $2/3$ vereinfacht, indem der Modul in den transformierten Modul N^T transformiert wird, wobei der transformierte Modul N^T größer als N ist, bevor irgendeine modulare Operation mit N ausgeführt wird. Dann werden alle Berechnungen Modulo N^T durchgeführt. Nachdem das Ergebnis der Berechnung jedoch in der Restklasse N sein muß, wird erfindungsgemäß eine abschließende Reduktion mit N durchgeführt.

10 Wie es in Fig. 2 gezeigt ist, sei N eine Ganzzahl mit einer Länge von N Bits. Da der Modul N immer eine positive Ganzzahl ist, d. h. $MSB = 0$ in der Zweier-Komplement-Darstellung, ist das Vorzeichenbit gleich 0 und das zweit-höchstwertige Bit (MSB -1) des Moduls N ist immer gleich 1. Für den ZDN-
15 Vergleich ist es nicht erforderlich, sämtliche Bits des Moduls mit sämtlichen Bits des Zwischenergebnisses zu vergleichen, sondern es ist ausreichend, eine Anzahl von m Bits für den ZDN-Vergleich zu verwenden. Die höchstwertigen m Bits des Moduls N definieren einen ersten Teil des Moduls N_T , während
20 die restlichen $N-m$ Bits des Moduls einen zweiten Teil N_R des Moduls definieren. Bei einem bevorzugten Ausführungsbeispiel ist m gleich 16. Selbstverständlich sind auch größere oder kleinere Werte von m möglich.

25 Wie es in Fig. 3 gezeigt ist, wird die Transformation derart ausgeführt, daß der transformierte Modul N^T 16 Bit länger ist als der ursprüngliche Modul von Fig. 2.

Für den ZDN-Vergleich ist es ausreichend, die ersten 16 Bit von N^T zu verwenden, wobei bei einem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung nur 12 Bits zum Vergleich verwendet werden, während die niederstwertigen 4 Bits einen Puffer für mögliche Überträge darstellen, die von noch niederwertigeren Bits kommen können.

35 In diesem Fall ist die Wahrscheinlichkeit, daß der Vergleich ein falsches Ergebnis ergibt, kleiner als 2^{-12} . Falls der Ver-

gleich ein falsches Ergebnis liefert, wird nur ein suboptimaler Reduktions-Verschiebungswert s_N erzeugt, das Ergebnis Modulo N ist jedoch nach wie vor korrekt.

- 5 Wenn der Modul wie in Fig. 2 in der Zweierkomplementdarstellung verwendet wird, dann kann der Modul N folgendermaßen zerlegt werden:

$$N = 2^{n-m} N_T + N_R.$$

10

Nun wird N zu N^T unter Verwendung des Transformators T transformiert, wobei T eine geeignet gewählte Ganzzahl ist, was aus Kongruenzgründen der Fall sein muß. N^T sollte die in Fig. 3 gezeigte Form haben, d. h. das höchstwertige Bit (MSB) von N^T muß gleich 0 sein, da N^T eine positive Ganzzahl sein soll.

15

Wie es nachfolgend ausgeführt wird, müssen das zweithöchstwertige und das dritthöchstwertige Bit des transformierten Moduls gleich 1 sein, während sämtliche anderen Bits des obersten Abschnitts des transformierten Moduls N^T , welcher in Fig. 3 mit dem Bezugszeichen 33 bezeichnet ist, einen Wert von „0“ haben sollten. Nur in diesem Fall ergibt sich nämlich für das 2/3-fache von N^T , daß der oberste Abschnitt des 2/3-fachen von N^T , wie es in Fig. 4 gezeigt ist, lediglich ein Bit mit einer „1“ hat, während alle anderen Bits in diesem obersten Abschnitt 44 gleich „0“ sind, so daß der bereits beschriebene triviale Vergleich zur Bestimmung von s_i ausgeführt werden kann.

20

25

Zunächst wird jedoch anhand von Fig. 3 auf die Berechnung des transformierten Moduls N^T unter Verwendung des Transformators T eingegangen. Es gelte folgende Definition:

30

$$N^T = T N$$

35

$$= T(2^{n-m} N_T + N_R)$$

Für den Transformator T gilt folgendes:

$$T = \left\lfloor \frac{2^{p-2} + 2^{p-3}}{N_T} \right\rfloor$$

5

Unter Verwendung von Gleichung 17 ergibt sich für den transformierten Modul N^T folgendes:

10

$$N^T = \left\lfloor \frac{2^{p-2} + 2^{p-3}}{N_T} \right\rfloor (2^{n-m} N_T + N_R)$$

15

$$N^T = (2^{n+p-m-2} + 2^{n+p-m-3}) \frac{N_T}{N_T} + (2^{p-2} + 2^{p-3}) \frac{N_R}{N_T}.$$

20 Wenn beispielsweise typische Werte für p und m genommen werden, also p gleich 32 Bit und m gleich 16 Bit, so ergibt sich für N^T folgendes:

$$N^T = 2^{n+14} + 2^{n+13} + N_R \frac{2^{p-2} + 2^{p-3}}{N_T}.$$

Es sei darauf hingewiesen, daß die Berechnung von N^T vorzugsweise in der Host-CPU durchgeführt wird, und nicht im Kryptocoprozessor. Die Host-CPU umfaßt ein Kurzzahl-Rechenwerk, was jedoch für die Berechnung von N^T ausreichend ist. Da T eine Ganzzahl sein muß und die Berechnungen innerhalb des Kryptocoprozessors Modulo N^T anstatt Modulo N durchgeführt werden, wobei N^T größer als N ist, sind nur die ersten $p-m$ gleich 16 Bits von N^T für den trivialen ZDN-Vergleich, um den Hilfsverschiebungswert s_i zu berechnen, relevant. Die anderen n Bits von N^T können irgendeine Zahl sein, sie sind für die Be-

rechnung des Hilfs-Verschiebungswerts s_i , also für den Vergleich mit Z nicht relevant. Selbstverständlich werden jedoch alle Bits des transformierten Moduls N^T für die Drei-Operanden-Addition benötigt, die nunmehr statt unter Verwendung des verschobenen Moduls unter Verwendung des verschobenen transformierten Moduls ausgeführt wird.

Wie es in Fig. 17 dargestellt ist, ist für die gewählten Werte für m und p der Transformator T eine 16-Bit-Ganzzahl. Daher muß die Division, die zur Berechnung von T erforderlich ist, bzw. die zur Berechnung von N^T erforderlich ist, nur für die höchstwertigen 32 Bits durchgeführt werden, und kann daher schnell und einfach auf der Host-CPU programmiert werden.

In Fig. 4 ist das 2/3-fache des transformierten Moduls N^T gezeigt. Da das MSB-1 und das MSB-2 von N^T gleich „1“ sind, wie es in Fig. 3 gezeigt ist, und folgendes gilt:

$$(11)_2 = (3)_{10} \text{ und } (2/3 \times 3)_2 = (2)_{10} = (10)_2,$$

ergibt sich ein einfaches Bitmuster für das 2/3-fache des transformierten Moduls N^T , wobei die Länge des 2/3-fachen des transformierten Moduls N^T gleich $n-m+p$ ist.

Aufgrund der speziellen Gestalt von $2/3 N^T$ wird nun der Vergleich mit $|Z|$ sehr einfach. Es ist bekannt, daß die höchstwertige Eins von $2/3 N^T$ an einer Position $n+p-m-2$ an dem Beginn einer modularen Operation ist. Ein Zeiger für das Register Z startet dann bei einem bevorzugten Ausführungsbeispiel an dem MSB von Z und sucht nach der ersten „1“ von Z . Wenn das MSB von Z gleich 1 ist, dann ist Z eine negative Zahl, und man sucht statt dessen die erste Null von Z .

Die Differenz der Bitposition der ersten Eins im Register N und im Register Z bestimmt den Hilfs-Verschiebungswert s_i .

Da das Ergebnis der Modulo-Operation in der Restklasse N sein muß, wird erfindungsgemäß eine Endreduktion Modulo N durchgeführt, es muß also eine Rücktransformation (Schritt 14 in Fig. 1) durchgeführt werden.

5

Die Transformation von N zu N^T hat die folgenden Vorteile im Vergleich zum bekannten ZDN-Vergleich:

10

Statt der Berechnung von $2/3 N$ innerhalb des Kryptocoprozessors kann eine einfache Transformation von N in N^T in der Host-CPU durchgeführt werden.

15

Auf dem Chip werden kein ZDN-Register und keine Komparatorlogik benötigt, weshalb die Chipgröße kleiner und die Komplexität des Coprozessors geringer werden.

20

Schließlich kann die Transformation von N in N^T mit einer Randomisierung des Moduls N kombiniert werden, wie es anhand von Fig. 5 dargestellt ist. Wenn R eine s Bit lange zufällige Zahl ist, hat der randomisierte transformierte Modul N^T die in Fig. 5 gezeigte Form. Durch die Randomisierungszahl N wird der randomisierte transformierte Modul im Vergleich zu dem Fall, bei dem keine Randomisierung durchgeführt worden ist (Fig. 3) um s Bit länger, also um die Anzahl der Stellen von R .

Gleichungsmäßig läßt sich dies folgendermaßen ausdrücken:

30

$$\begin{aligned} N^T &= T N \\ &= T(2^{n-m} N_T + N_R) \end{aligned}$$

Der randomisierte Transformator T lautet dann folgendermaßen:

35

$$T = \left| \frac{2^{p-2} - 2^{p-3} + R}{N_T} \right|$$

Damit ergibt sich folgender Ausdruck für den randomisierten transformierten Modul:

5

$$N^T = \left\lfloor \frac{2^{p-2} + 2^{p-3} + R}{N_T} \right\rfloor (2^{n-m} N_T + N_R)$$

10

$$N^T = (2^{n+p-m-2} + 2^{n+p-m-3} + R \cdot 2^{n-m}) \frac{N_T}{N_T} + (2^{p-2} + 2^{p-3} + R) \frac{N_R}{N_T}.$$

Wenn für p gleich 144 Bits, m gleich 16 Bit und s gleich 112 Bit eingesetzt werden, ergibt sich für den transformierten Modul N^T einschließlich Randomisierung folgender Wert:

15

$$N^T = 2^{n+126} + 2^{n+125} + R \cdot 2^{n-16} + N_R \frac{2^{144} + 2^{143} + R}{N_T}.$$

20

Die Bitlänge von N^T ist dann:

$$L(N^T) = n+p-m = n+m+s = n+16+112 = n+128 \text{ Bits}$$

Fig. 6 zeigt ein erfindungsgemäßes Rechenwerk, das im Vergleich zu Fig. 7 nunmehr kein ZDN-Register hat, sondern lediglich noch eine arithmetische Einheit 700, ein C-Register 710, ein N-Register 720 und ein Z-Register 730, wobei im N-Register 720 nun nicht mehr der Modul bzw. ein verschobener Modul gespeichert ist, sondern der transformierte Modul bzw. ein verschobener transformierter Modul oder aber ein randomisierter transformierter Modul oder ein verschobener randomisierter transformierter Modul.

35

Im nachfolgenden wird auf die Fig. 8a bis 8c eingegangen, um den Zusammenhang zwischen dem Hilfs-Verschiebungs-Wert s_i und dem Reduktions-Verschiebungswert s_N darzustellen.

5 In nachfolgenden wird auf die Fig. 8a bis 8c eingegangen, um die Berechnung des Reduktions-Verschiebungswerts s_z unter Verwendung des Hilfs-Reduktions-Verschiebungswerts s_i darzustellen. In Fig. 8a sind ein Zwischenergebnis Z und ein Modul N dargestellt. Lediglich beispielhaft hat das Zwischenergebnis
10 nis vier Bits, während der Modul 9 Bits hat. Nunmehr sei angenommen, daß in dem Block 214 von Fig. 2 ein verschobenes Zwischenergebnis Z' berechnet wird, was durch Multiplizieren mit s_z erreicht werden kann. So sei angenommen, daß im Multiplikator 8 Nullen waren, was dazu führt, daß der Multiplikations-Verschiebungswert s_z gleich 8 war. Um eine modulare Reduktion zu erreichen, muß der Modul N in die Größenordnung des verschobenen Zwischenergebnis Z' kommen. Erfindungsgemäß soll der Modul N so weit verschoben werden, daß das oberste Bit des verschobenen Zwischenergebnis-Polynoms Z' und das
15 oberste Bit des verschobenen Moduls N gleich sind. Wie es aus Fig. 8b zu sehen ist, ist hierzu ein Reduktions-Verschiebungswert von s_N gleich 3 erforderlich.

Aus Fig. 8b ist ebenfalls zu sehen, daß die Ermittlung von s_N eigentlich erst durchgeführt werden kann, wenn s_z berechnet worden ist, d. h. daß eine parallele Ausführung der Blöcke 210 und 212 von Fig. 2, wie es für die vorliegende Erfindung bevorzugt wird, nicht möglich ist. Aus diesem Grund wird der Hilfs-Verschiebungs-Parameter s_i eingeführt. Vorteilhaft an
30 s_i ist, daß dieser Wert berechnet werden kann, ohne das s_z des aktuellen Schritts zu kennen.

Aus Fig. 8c ist zu sehen, daß s_z immer gleich der Summe aus s_i und s_N ist. s_N hängt somit immer mit s_z und s_i derart zusammen,
35 daß folgende Gleichung gilt:

$$s_N = s_z - s_i.$$

Das zeitaufwendige iterative Verfahren zum Bestimmen von s_N kann somit zerlegt werden in ein zeitaufwendiges iteratives Verfahren zum Bestimmen von s_i (Schleife 416) und eine
5 schnelle Differenz-Operation (Block 422 von Fig. 4). Damit ist eine nahezu parallele Ausführung der beiden Vorausschau-Verfahren möglich, wobei die einzige serielle Komponente darin besteht, daß vor dem Berechnen des Blocks 422 (Fig. 4) der tatsächliche Wert von s_z durch den Multiplikations-
10 Vorausschau-Algorithmus bereits berechnet und geliefert worden ist (Pfeil 230 in Fig. 2).

Zusammenfassend sei ausgeführt, daß die vorliegende Erfindung den Vergleich zwischen $2/3 N$ und dem Betrag von Z im Vergleich zum bekannten ZDN-Verfahren vereinfacht. Im Gegensatz zum bisherigen bekannten Verfahren, bei dem die obersten 32 Bit von $2/3 N$ im Kryptocoprozessor berechnet und in einem separaten 32-Bit-Register, dem ZDN-Register, abgelegt wurden, wobei der Vergleich von $2/3 N$ mit dem Betrag von Z gemäß dem
15 bekannten ZDN-Verfahren in Hardware über einen Komparator ausgeführt wurde, der Bestandteil des Controlteils des Kryptocoprozessors war, wird nunmehr folgendermaßen vorgegangen. Der Modul N wird von der Host-CPU in einen transformierten Modul N^T transformiert, der größer als N ist, wobei die ersten Bits von N^T eine Konstante sind, die so gewählt ist, daß der Vergleich von $2/3 N^T$ mit dem Betrag von Z trivial ist. Zur Verbesserung der Sicherheit gegen Informations-Leck-Attacken, wie z. B. SPA, DPA, Timing-Attacken, ..., kann die Transformation von N zu N^T mit der Randomisierung des Moduls
20 kombiniert werden, wie es ausgeführt worden ist.
30

Damit entfällt die $2/3 N$ -Berechnung im Kryptocoprozessor. Das ZDN-Register und die Komparatorlogik entfallen ebenso, wodurch sich eine kleinere Chipfläche und eine Verringerung der
35 Komplexität des Controlteils im Kryptocoprozessor durch Wegfall der Komparatorlogik ergeben.

Patentansprüche

1. Verfahren zum modularen Multiplizieren eines Multiplikanden (C) mit einem Multiplikator (M), wobei ein Modul (N) verwendet wird, unter Verwendung eines Multiplikations-Vorausschau-Verfahrens und eines Reduktions-Vorausschau-Verfahrens, mit folgenden Schritten:

Transformieren (10) des Moduls (N) in einen transformierten Modul (N^T), der größer als der Modul (N) ist, wobei ein vorbestimmter Bruchteil ($2/3$) des transformierten Moduls eine höherwertige Stelle mit einem ersten vorbestimmten Wert hat, der zumindest eine niederwertigere Stelle folgt, die einen zweiten vorbestimmten Wert hat;

iteratives Abarbeiten (12) der modularen Multiplikation unter Verwendung des Multiplikations-Vorausschau-Verfahrens und des Reduktions-Vorausschau-Verfahrens unter Verwendung des transformierten Moduls (N^T), um am Ende der Iteration ein transformiertes Ergebnis für die modulare Multiplikation zu erhalten; und

Rücktransformieren (14) des transformierten Ergebnisses durch modulares Reduzieren des transformierten Ergebnisses unter Verwendung des Moduls (N).

2. Verfahren nach Anspruch 1, bei dem im Schritt des iterativen Abarbeitens (12) eine Mehrzahl von Iterationsschritten durchgeführt wird, wobei in einem Iterationsschritt ein Multiplikations-Zwischenergebnis und ein Reduktions-Verschiebungswert (s_N) ermittelt werden, wobei der Reduktions-Verschiebungswert (s_N) unter Verwendung einer Bestimmung der Anzahl (s_i) von Stellen zwischen der höherwertigen Stelle mit dem ersten vorbestimmten Wert des transformierten Moduls (N^T) und der höchstwertigen Stelle des Zwischenergebnisses (Z), die den ersten vorbestimmten Wert hat, berechnet wird.

3. Verfahren nach Anspruch 2, bei dem in dem Multiplikations-Vorausschau-Verfahren ein Multiplikations-Verschiebungswert (s_z) ermittelt wird, und bei dem ein Reduktions-Verschiebungswert (s_N) für das Reduktions-Vorausschau-

5 Verfahren durch Subtrahieren der bestimmten Anzahl (s_i) von Stellen von dem Multiplikations-Verschiebungswert (s_z) berechnet wird.

10 4. Verfahren nach einem der vorhergehenden Ansprüche, bei dem der Schritt des iterativen Abarbeitens folgende Schritte aufweist:

in einem ersten Iterationsschritt:

15 (a) Durchführen eines Multiplikations-Vorausschau-Verfahrens, um einen Multiplikations-Verschiebungswert (s_z) zu erhalten,

(b) Multiplizieren einer Basis potenziert mit dem Multiplikations-Verschiebungswert mit einem aktuellen Zwischenergebnis
20 (Z), um ein verschobenes Zwischenergebnis (Z') zu erhalten;

(c) Durchführen eines Reduktions-Vorausschau-Verfahrens, um einen Reduktions-Verschiebungswert (s_N) zu erhalten durch Bestimmen eines Hilfs-Verschiebungswerts (s_i) der gleich der Anzahl von Stellen zwischen der höherwertigen Stelle mit dem ersten vorbestimmten Wert des vorbestimmten Bruchteils des transformierten Moduls (N^T) und der höchstwertigen Stelle des Zwischenergebnisses, die den ersten vorbestimmten Wert hat, ist, und durch Berechnen des Reduktions-Verschiebungswerts
30 unter Verwendung des Hilfs-Verschiebungswerts und des Multiplikations-Verschiebungswerts (s_z);

(d) Multiplizieren des transformierten Moduls (N^T) mit der Basis potenziert mit dem Reduktions-Verschiebungswert, um einen verschobenen transformierten Modul ($N^{T'}$) zu erhalten; und
35

(e) Summieren des Zwischenergebnisses (Z') und des Multiplikanden (C) und Subtrahieren des verschobenen transformierten Moduls (N^T), um ein aktualisiertes Zwischenergebnis (Z) zu erhalten.

5

5. Verfahren nach einem der vorhergehenden Ansprüche, bei dem der vorbestimmte Bruchteil des Moduls $2/3$ beträgt.

10

6. Verfahren nach Anspruch 5, bei dem der Multiplikand (C), der Multiplikator (M) und der Modul (N) binär sind, bei dem die Basis 2 beträgt, und bei dem die höherwertige Stelle des vorbestimmten Bruchteils des transformierten Moduls (N^T) einen ersten vorbestimmten Wert von 1 hat, und die zumindest eine niederwertige Stelle einen zweiten vorbestimmten Wert von 0 hat.

15

7. Verfahren nach Anspruch 6, bei dem das höchstwertige Bit des transformierten Moduls ein Vorzeichen-Bit ist und ein höherwertiger Abschnitt des vorbestimmten Bruchteils des Moduls folgendermaßen lautet:

20

01000 xx ... xx,

wobei die mit xx bezeichneten Bits beliebige Werte haben können.

8. Verfahren nach Anspruch 7, bei dem der höherwertige Abschnitt des transformierten Moduls (N^T) folgendermaßen lautet:

30

01100 ... 00.

9. Verfahren nach einem der Ansprüche 1 bis 8, bei dem im Schritt des Transformierens (10) des Moduls eine Randomisierung des Moduls durchgeführt wird, so daß der transformierte Modul randomisiert ist.

35

10. Prozessor zum modularen Multiplizieren eines Multiplikanden (C) mit einem Multiplikator (M), wobei ein Modul (N) verwendet wird, unter Verwendung eines Multiplikations-Vorausschau-Verfahrens und eines Reduktions-Vorausschau-

5 Verfahrens, mit folgenden Merkmalen:

einer Einrichtung zum Transformieren (10) des Moduls (N) in einen transformierten Modul (N^T), der größer als der Modul (N) ist, wobei ein vorbestimmter Bruchteil des transformierten Moduls eine höherwertige Stelle mit einem ersten vorbestimmten Wert hat, der zumindest eine niederwertigere Stelle
10 folgt, die einen zweiten vorbestimmten Wert hat;

einer Einrichtung zum iterativen Abarbeiten (12) der modularen Multiplikation unter Verwendung des Multiplikations-Vorausschau-Verfahrens und des Reduktions-Vorausschau-Verfahrens unter Verwendung des transformierten Moduls (N^T), um am Ende der Iteration ein transformiertes Ergebnis für die modulare Multiplikation zu erhalten; und
15

20

einer Einrichtung zum Rücktransformieren (14) des transformierten Ergebnisses durch modulares Reduzieren des transformierten Ergebnisses unter Verwendung des Moduls (N).

25

11. Prozessor nach Anspruch 10, der eine Host-CPU und einen Coprozessor aufweist, wobei die Einrichtung zum Transformieren (10) des Moduls in der Host-CPU angeordnet ist, und wobei die Einrichtung zum iterativen Abarbeiten (12) der modularen Multiplikation in dem Coprozessor angeordnet ist.

30

12. Prozessor nach Anspruch 11, bei dem die Host-CPU ein Kurzzahl-Rechenwerk mit einer Anzahl von Stellen kleiner oder gleich 64 aufweist, und bei dem der Coprozessor ein Langzahl-Rechenwerk mit einer Anzahl von Stellen größer oder gleich
35 512 aufweist.

13. Prozessor nach einem der Ansprüche 10 bis 12,

bei dem die Einrichtung zum iterativen Abarbeiten der modularen Multiplikation ein Register für den transformierten Modul und ein Register für ein Zwischenergebnis der modularen Multiplikation aufweist.

5

Zusammenfassung

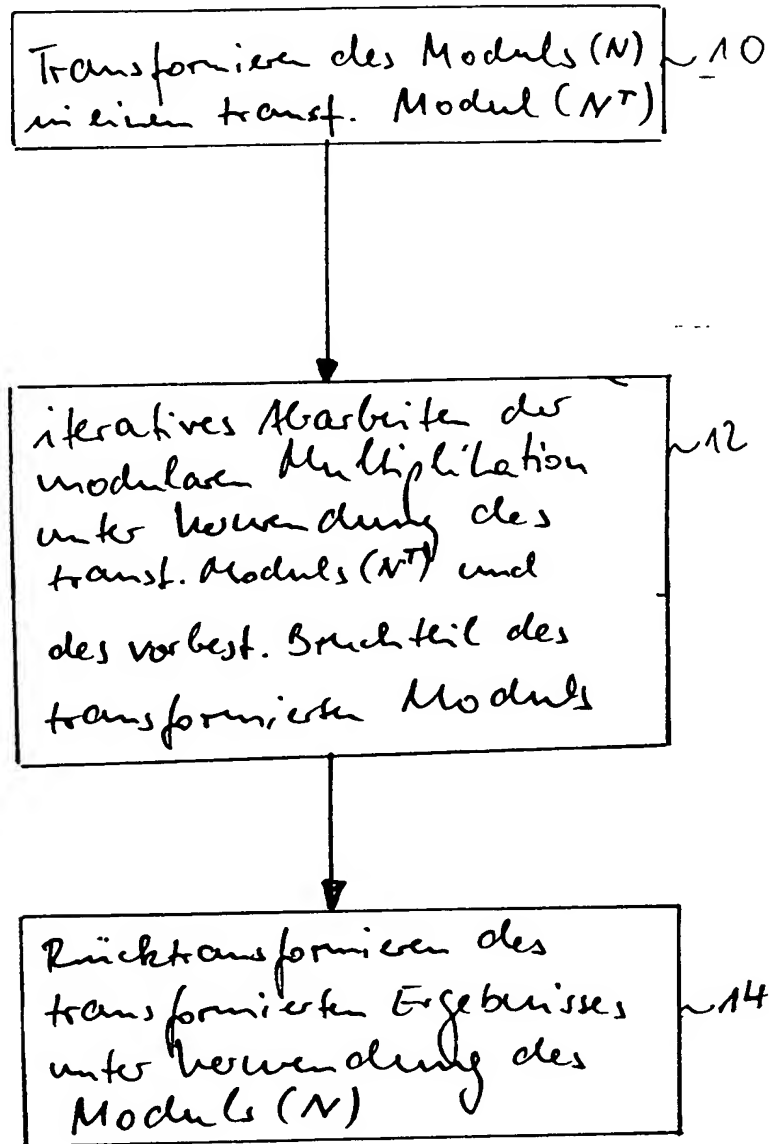
Verfahren und Vorrichtung zum modularen Multiplizieren

5 Bei einem Verfahren zum modularen Multiplizieren unter Verwendung eines Multiplikations-Vorausschau-Verfahrens zum Berechnen eines Multiplikations-Verschiebungswerts und eines Reduktions-Vorausschau-Verfahrens zum Berechnen eines Reduktions-Verschiebungswerts wird ein Modul zunächst in einen
10 transformierten Modul transformiert (10), der größer als der Modul ist. Die Transformation wird so durchgeführt, daß ein vorbestimmter Bruchteil des transformierten Moduls eine höherwertige Stelle mit einem ersten vorbestimmten Wert hat, dem zumindest eine niederwertige Stelle folgt, die einen
15 zweiten vorbestimmten Wert hat. Während des iterativen Abarbeitens (12) der modularen Multiplikation unter Verwendung des Multiplikations-Vorausschau-Verfahrens und des Reduktions-Vorausschau-Verfahrens wird der transformierte Modul verwendet, um am Ende der Iteration ein transformiertes Ergebnis
20 für die modulare Multiplikation zu erhalten. Schließlich wird das transformierte Ergebnis durch modulares Reduzieren unter Verwendung des ursprünglichen Moduls rücktransformiert (14). Durch die erfindungsgemäße Transformation wird das iterative Abarbeiten der modularen Multiplikation vereinfacht, so daß
25 die modulare Multiplikation schneller durchgeführt werden kann.

30

Figur 1

Figur für die Zusammenfassung



Bezugszeichenliste

- 10 Transformieren des Moduls
- 12 Iteratives Abarbeiten der modularen Multiplikation
- 14 Rücktransformieren des transformierten Ergebnisses
- 33 Oberer Abschnitt des transformierten Moduls
- 44 Oberer Abschnitt des 2/3-fachen des transformierten Moduls
- 700 Arithmetische Einheit
- 710 C-Register
- 720 N-Register
- 730 Z-Register
- 740 Iterationsschleife
- 750 ZDN-Register
- 760 Steuerlogik für den ZDN-Vergleich
- 900 Start des ZDN-Verfahrens
- 910 Multiplikations-Vorausschau-Verfahren für den ZDN-Algorithmus
- 920 Verschieben von Z nach links oder rechts
- 930 Reduktions-Vorausschau-Verfahren für den ZDN-Algorithmus
- 940 Verschieben des Moduls nach links
- 950 Drei-Operanden-Addition für den ZDN-Algorithmus
- 960 Ende des ZDN-Algorithmus
- 1000 Globale Variablen
- 1010 Initialisierung des Reduktions-Verschiebungswerts
- 1020 Berechnen von ZDN
- 1030 Untersuchen von n und s_N
- 1040 Dekrementieren von n
- 1060 Dekrementieren des Reduktions-Verschiebungswerts
- 1080 Berechnen von ZDN/2
- 1100 Vergleich des Zwischenergebnisses mit ZDN
- 1120 Bestimmen des Reduktions-Vorausschau-Parameters
- 1140 Bestimmen des Reduktions-Vorausschau-Parameters
- 1160 Berechnen von n
- 1180 Berechnen des Reduktions-Vorausschau-Parameters
- 1200 Untersuchen von n

1220 Berechnen von cur_k

1240 Berechnen von cur_k

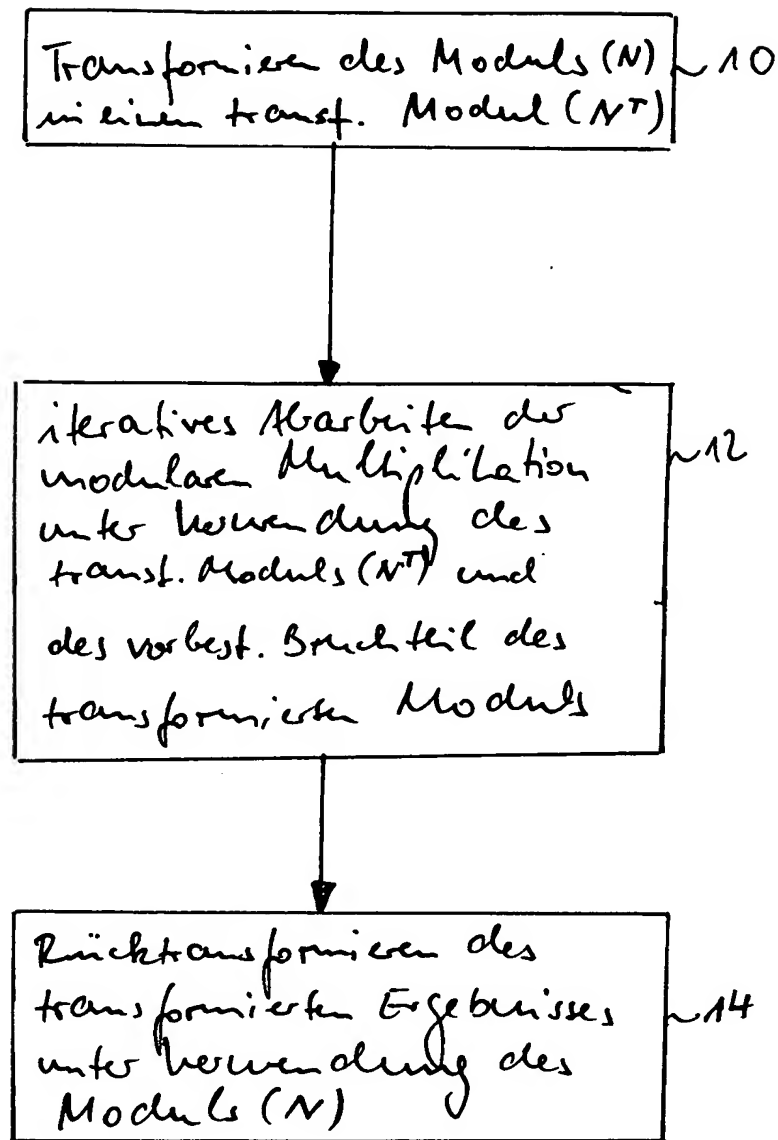


Fig. 1

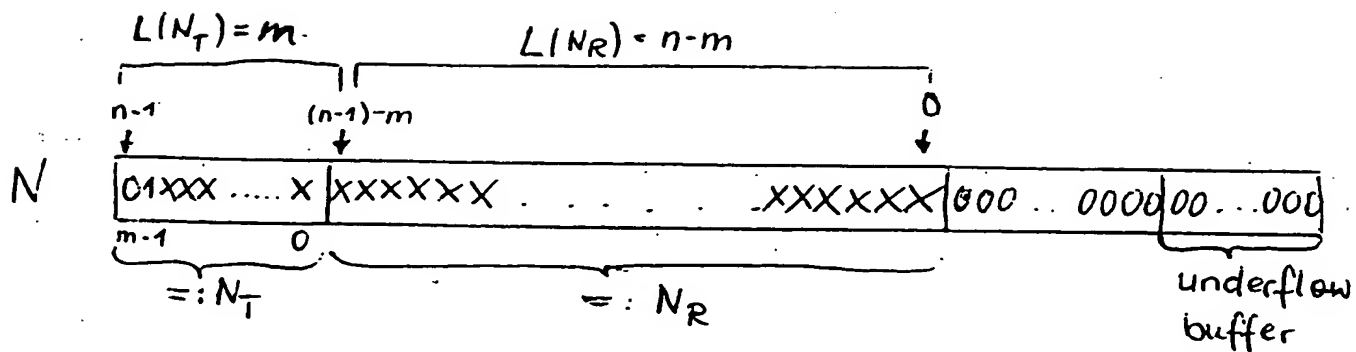


Fig. 2 (Modul)

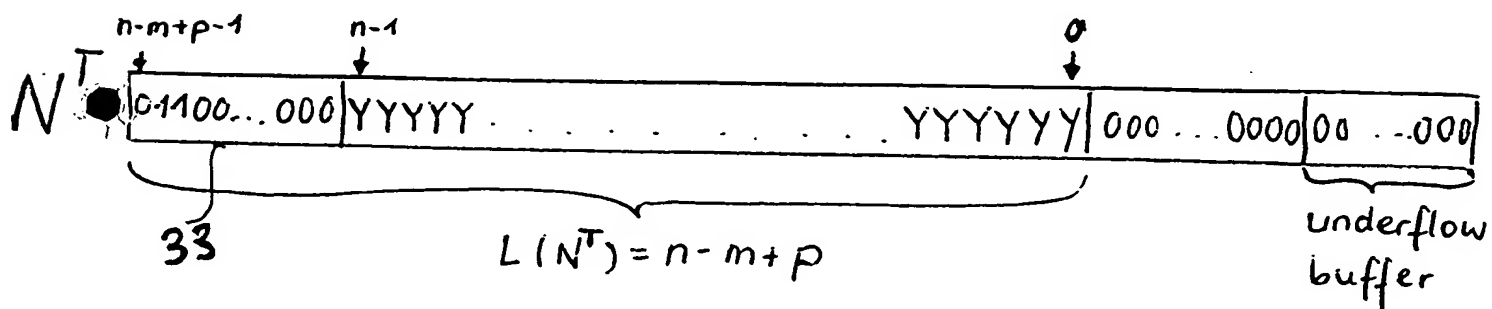


Fig. 3 (transformierter Modul)

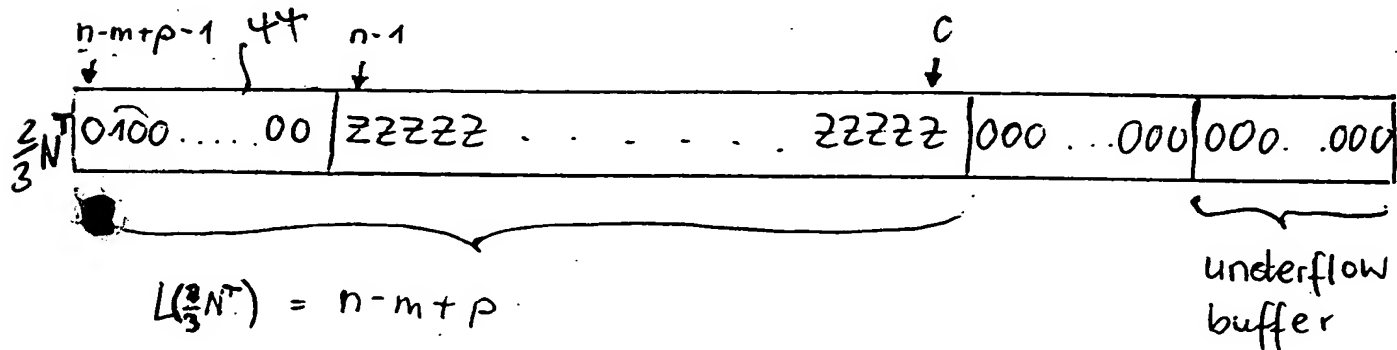


Fig. 4 ($\frac{2}{3}N^T$)

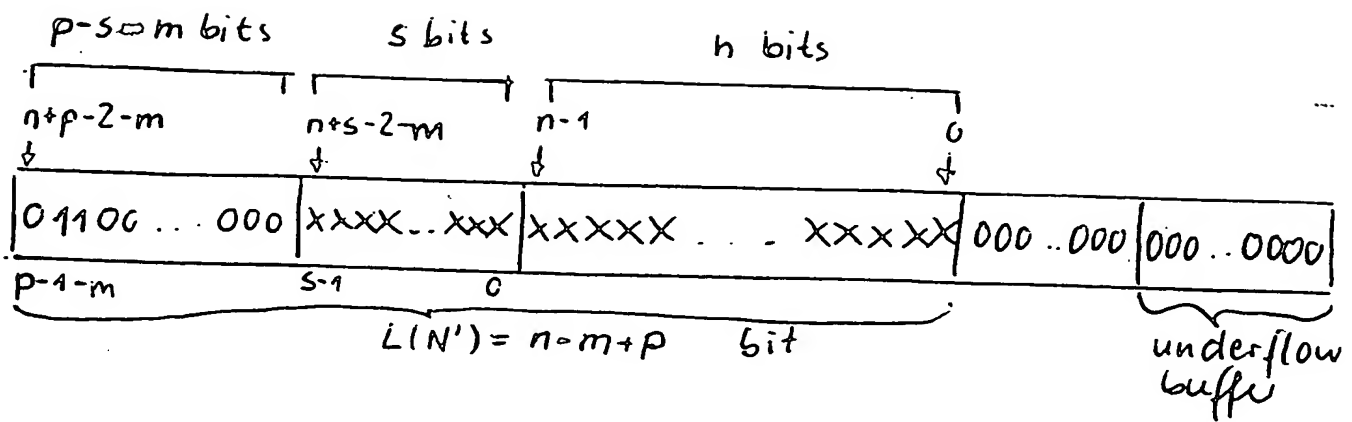


Fig. 5 (transformierter Modul
mit Randomisierung)

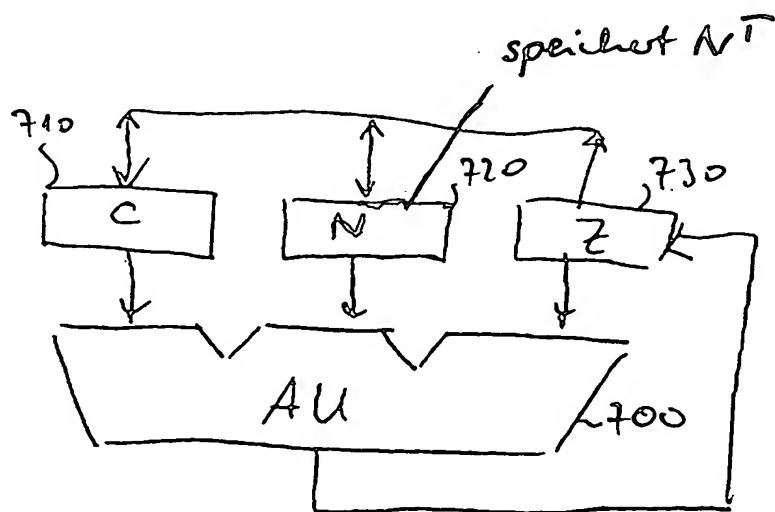


Fig. 6

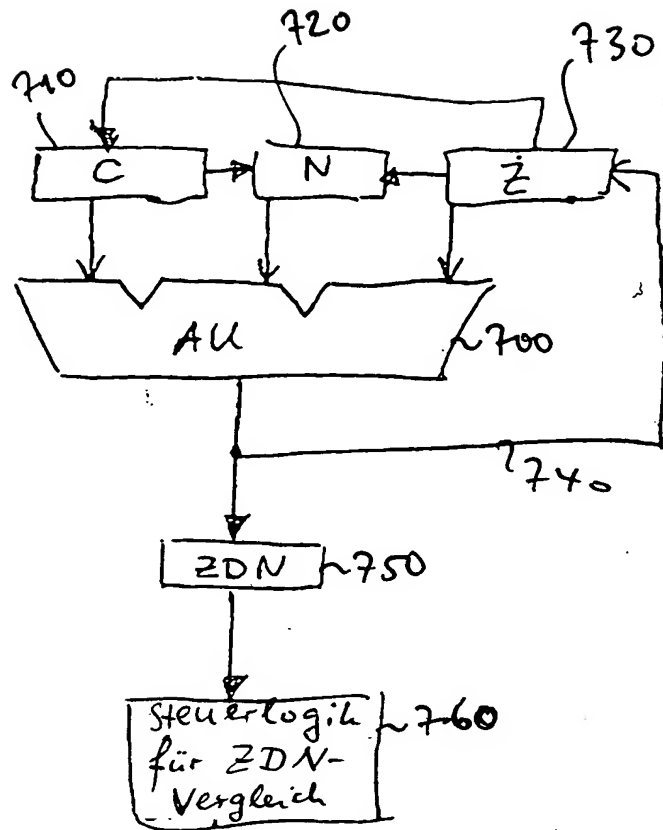


Fig. 7 (Stand der Technik)

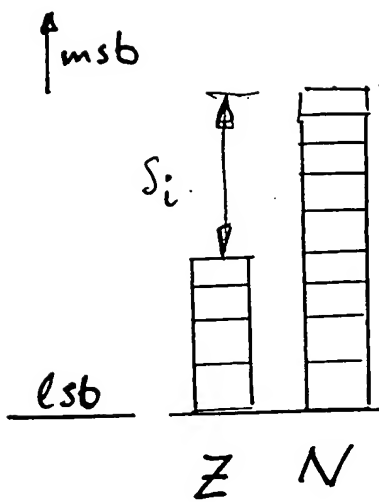


Fig. 8a

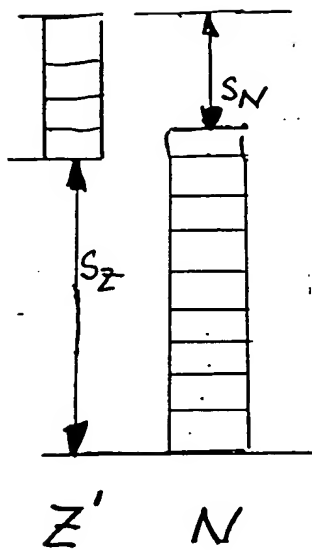


Fig. 8b

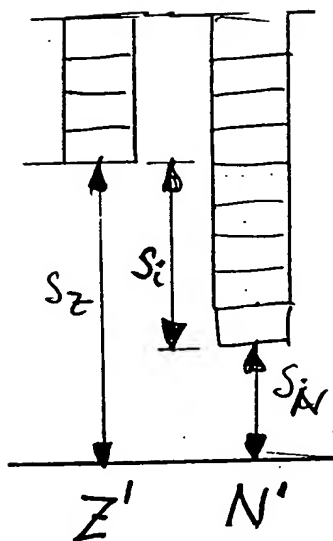


Fig. 8c

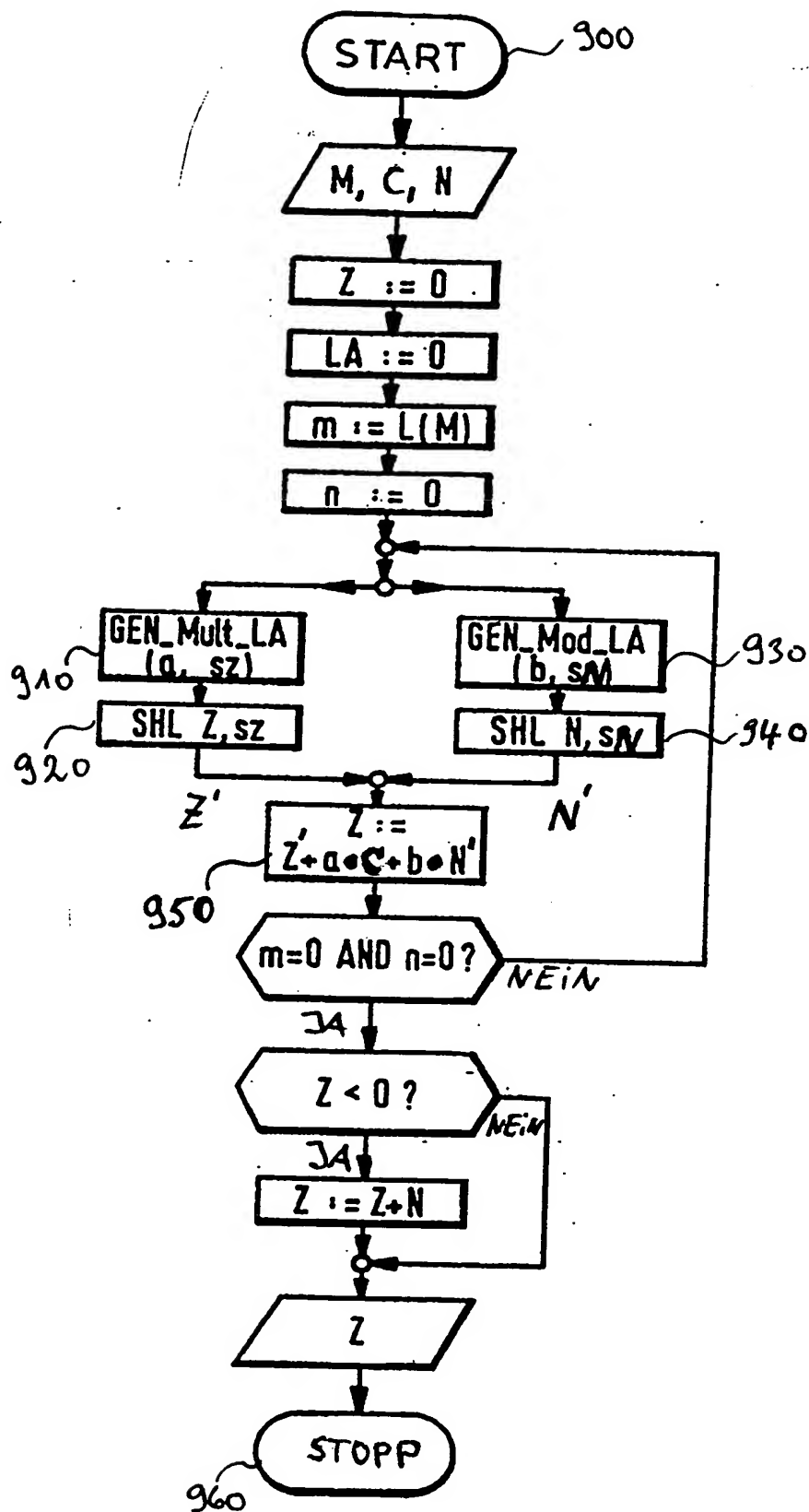


Fig. 9 (Stand der Technik)

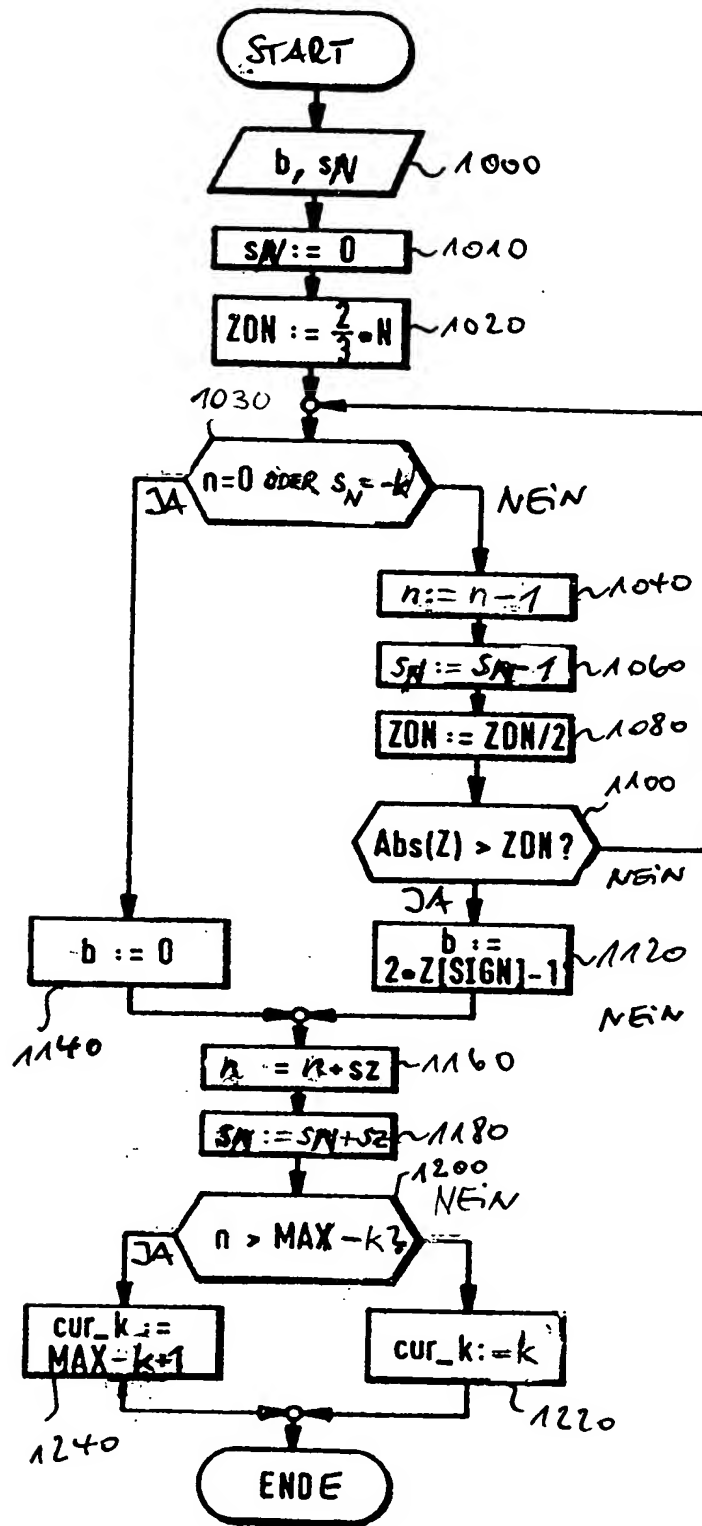


Fig. 10 (Stand der Technik)